# Goal

The type of malware we develop depends completely on the goal of the mission. If the goal is to hijack web-traffic of a users that use banks it might be better to write the malware to disk, and have it execute on boot and then lay silent until the suer authenticates to his bank. If the goal is to encrypt the entire disk, well then that is what the malware will do. If the goal is gain an entrance to network, it might be useful with a shell. Do you want it to spread automatically, like a virus? Then you need those functionalities. Do you want it to be more like a backdoor? So you get a shell whenever you want.

We want to build a Backdoor, more than a specific malware.

1. We want to write as little to disk as possible. Instead of writing a payload to disk we will activate it on boot, and the dropper will be executed everytime. This is good because if the malware is discovered it will not reveal the actual payload, only the dropper. This is assuming that the target is a user that produces a lot of network traffic, so another request will not make much of a differece.
2. We want persistence.
3. We want a shell. Simple as that.

## First stage loader - Dropper

The idea is to have a lightwight first stage payload. The first stage payload will only load in the second stage payload in memory.

See the next chapter on how to generate Droppers.

## Ways too load into memory - Fileless stager

The traditional way has been to download the second stager to disk somewhere. But new research has found more efficient ways to do this, not having to write anything to disk. The new way to only load into memory is somtimes calld a fileless stager. - Using PowerShell This is the most common way. And we have all used it, invoke-expression.

- Using WMI
- PreProcessor
-

## Persistence

### WMI

WMI is often used. You can create events that are executed when the event is triggered. An event can be anything, like when the screensaver runs, when the computer is locked. It requires Administrative priviliges in order to create a permanent event subscription.

You can also create a new WMI class. And in that class store a base64-encrypted property, which is in fact a powershell command.

Then you create a WMI event subscripting. When the event is triggered the base64-encoded property is retrieved and the powershell command is excecuted.

This way the only backdoor that is left is in the WMI repository, nothing else on disk.

This was documented here: https://www.fireeye.com/blog/threat-research/2017/03/dissecting_one_ofap.html There are a few ways https://blog.netspi.com/getting-started-wmi-weaponization-part-5/